Sécurisation des communications

Exercice 1

Dans cet exercice on se propose d'écrire les fonctions Python correspondant au chiffrement de César. On suppose que les messages sont composés uniquement de majuscules qui seront chiffrées et d'espaces qu'on ne modifie pas.

1. Écrire en Python une fonction `chiffre_cesar(msg, cle)` qui prend en arguments une chaîne de caractères `msg` et un entier `clé` et renvoie une chaîne de caractère correspondant au chiffrement de César de la chaîne `msg` avec un décalage égal à `clé`.

```
>>> chiffre_cesar("L INFORMATIQUE C EST SUPER",5)
'Q NSKTWRFYNVZJ H JXY XZUJW'
```

2. Écrire une fonction Python `dechiffre_cesar(msgc, cle)` qui réalise le déchiffrement du message chiffré `msgc` avec le code de César pour un décalage égale à `clé`.

```
>>> dechiffre_cesar("Q NSKTWRFYNVZJ H JXY XZUJW",5)
'L INFORMATIQUE C EST SUPER'
```

3. En utilisant la fonction `chiffre_cesar`, écrire un programme Python qui essaye toutes les clés possibles et retrouver alors le message d'origine correspondant au message chiffré :

```
'QNAF Y NEVGUZRGVDHR QR Y NZBHE HA CYHF HA RTNYR GBHG RG QRHK ZBVAF HA RTNYR EVRA'
```

Indications :

- La fonction `ord` permet d'avoir l'unicode correspondant à un caractère.

```
>>> ord('A')
65
```

- La fonction `chr` permet d'avoir le charactère correspondant à un unicode donné.

```
>>> chr(65)
'A'
```

- L'opérateur « `a%b` » permet de calculer le reste de la division euclidienne de `a` par `b`.

```
>>> 30%26
4
```

Exercice 2

Écrire une fonction Python `chiffre_xor(msg, cle)` qui prend en argument deux chaînes d'octets (type `bytes`) et qui renvoie le chiffrement XOR du message avec la clé, sous la forme d'une chaîne d'octets.

```
>>> msg = "L'informatique c'est super!".encode()
>>> cle = "NSI".encode()
>>> msgc = chiffre_xor(msg,cle)
>>> msgc
b"\x02t  5&<>(::8;6i-t,='i=&9+!h"
>>> chiffre_xor(msgc,cle).decode()
"L'informatique c'est super!"
```

Indications :

- Une chaîne d'octets ressemble à une chaîne de caractères mais peut être considérée comme une liste d'entiers. Autrement dit, le parcours d'une chaîne d'octets est le parcours des entiers qui sont les codes des caractères de la chaîne.
- La méthode `encode()` retourne la chaîne d'octets correspondant à la chaîne de caractères sur laquelle elle est appelée.
- Inversement, la méthode `decode()` retourne (si possible) la chaîne de caractères correspondant à la chaîne d'octets sur laquelle elle est appelée.
- En principe il faut étendre la clé en la recopiant assez de fois jusqu'à obtenir la taille du message à coder/décoder. Cependant, un usage pertinent du reste de la division euclidienne permet de l'éviter.
- Le XOR entre deux entiers `a` et `b` se fait en Python avec la syntaxe « `a^b` ».
- Le constructeur `bytes` crée une chaîne d'octets à partir d'une liste d'entiers.

```
>>> L = [32, 66, 97, 195, 169]
>>> bytes(L)
b' Ba\xc3\xa9'
```

  On pourra alors construire la liste des entiers du message codé, puis transformer celle-ci en la chaîne d'octets à retourner.

Exercice 3

Soit la chaîne d'octets chiffrée à l'aide du chiffrement XOR :

```
b"\x0e6/+y;.< x-(7,,\x9b\xf0z48z:646<z*\x9a\xf3(64+<{"
```

On sait que les 4 derniers caractères du message en clair sont `"nse!"`. On sait aussi que la clé fait exactement 3 caractères et que ce sont des lettres majuscules sans accent.
Écrire un programme Python, utilisant la fonction `chiffre_xor` de l'exercice précédent, qui essaye toutes les combinaisons de clé jusqu'à trouver la bonne.
Mesurer le temps d'exécution. Pour cela, on pourra utiliser la fonction `time()` du module `time` (faire la différence de deux valeurs de retours de ces fonctions donne une durée en secondes).

Exercice 4

Écrire une fonction `factorisation(n)` qui prend en paramètre un entier `n` et renvoie un couple d'entiers `(p,q)` tel que `n=p*q` avec `1<p<=q` si `n` n'est pas premier et `p=1`, `q=n` si `n` est premier.
Trouver alors deux nombres `p` et `q` :

- tels que `p*q = 906555947934709` ;
- puis tels que `p*q = 17063866208590147`.

Mesurer le temps d'exécution.

Exercice 5

Écrire une fonction `log_discret(gu, g, p)` qui détermine l'entier `u` tel que le reste de la division euclidienne de g^u par `p` soit égale à `gu`.
Mesurer alors le temps d'exécution des appels suivants :

```
>>> log_discret(273213231, 7, 934741963)
>>> log_discret(360223736, 7, 934741963)
```

Indication : Pour calculer le reste de la division euclidienne de g^u par `p` on pourra utiliser `pow(g, u, p)`.

Exercice 6

Télécharger le certificat du site https://www.education.gouv.fr/.
Utiliser la commande `openssl` pour déterminer les informations de ce certificat.

Exercice 7

1. Qu'est ce qu'un certificat X.509 et quelles sont les principales informations qu'il contient ?
2. Discuter les scénarios suivants en termes de sécurité :
   (a) Deux certificats différents sont signés par la même clé privée.
   (b) Deux certificats différents contiennent la même clé publique.
   (c) Deux certificats différents ont le même sujet.
   (d) Deux certificats différents ont le même émetteur.